

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
APPLICATION FOR PATENT

Title: FLASH EEprom SYSTEM

Inventors: Eliyahou Harari, Robert D. Norman,
Sanjay Mehrotra

clm 91 >
Background of the Invention

5 This invention relates generally to semiconductor electrically erasable programmable read only memories (EEprom), and specifically to a system of integrated circuit Flash EEprom chips.

10 Computer systems typically use magnetic disk drives for mass storage of data. However, disk drives are disadvantageous in that they are bulky and in their requirement for high precision moving mechanical parts. Consequently they are not rugged and are prone to reliability problems, as well as consuming significant
15 amounts of power. Solid state memory devices such as DRAM's and SRAM's do not suffer from these disadvantages. However, they are much more expensive, and require constant power to maintain their memory (volatile). Consequently, they are typically used as
20 temporary storage.

EEprom's and Flash EEprom's are also solid state memory devices. Moreover, they are nonvolatile, and retain their memory even after power is shut down. However, conventional Flash EEprom's have a limited
25 lifetime in terms of the number of write (or program)/erase cycles they can endure. Typically the devices are rendered unreliable after 10^2 to 10^3 write/erase cycles. Traditionally, they are typically used in applications where semi-permanent storage of
30 data or program is required but with a limited need for reprogramming.

0899498-12297
66221-8646680

Accordingly, it is an object of the present invention to provide a Flash EEPROM memory system with enhanced performance and which remains reliable after enduring a large number of write/erase cycles.

5 It is another object of the present invention to provide an improved Flash EEPROM system which can serve as non-volatile memory in a computer system.

It is another object of the present invention to provide an improved Flash EEPROM system that can
10 replace magnetic disk storage devices in computer systems.

It is another object of the present invention to provide a Flash EEPROM system with improved erase operation.

15 It is another object of the present invention to provide a Flash EEPROM system with improved error correction.

It is yet another object of the present invention to provide a Flash EEPROM with improved write
20 operation that minimizes stress to the Flash EEPROM device.

It is still another object of the present invention to provide a Flash EEPROM system with enhanced write operation.

25 Summary of the Invention

These and additional objects are accomplished by improvements in the architecture of a system of EEPROM chips, and the circuits and techniques therein.

30 According to one aspect of the present invention, an array of Flash EEPROM cells on a chip is organized into sectors such that all cells within each sector are erasable at once. A Flash EEPROM memory system comprises one or more Flash EEPROM chips under the control of a controller. The invention allows any

465221-8646680

0899498 "122997
/65227" 8645680

combination of sectors among the chips to be selected and then erased simultaneously. This is faster and more efficient than prior art schemes where all the sectors must be erased every time or only one sector at a time can be erased. The invention further allows any combination of sectors selected for erase to be deselected and prevented from further erasing during the erase operation. This feature is important for stopping those sectors that are first to be erased correctly to the "erased" state from over erasing, thereby preventing unnecessary stress to the Flash EEPROM device. The invention also allows a global de-select of all sectors in the system so that no sectors are selected for erase. This global reset can quickly put the system back to its initial state ready for selecting the next combination of sectors for erase. Another feature of the invention is that the selection is independent of the chip select signal which enables a particular chip for read or write operation. Therefore it is possible to perform an erase operation on some of the Flash EEPROM chips while read and write operations may be performed on other chips not involved in the erase operation.

According to another aspect of the invention, improved error correction circuits and techniques are used to correct for errors arising from defective Flash EEPROM memory cells. One feature of the invention allows defect mapping at cell level in which a defective cell is replaced by a substitute cell from the same sector. The defect pointer which connects the address of the defective cell to that of the substitute cell is stored in a defect map. Every time the defective cell is accessed, its bad data is replaced by the good data from the substitute cell.

Another feature of the invention allows defect mapping at the sector level. When the number of

4

defective cells in a sector exceeds a predetermined number, the sector containing the defective cells is replaced by a substitute sector.

5 An important feature of the invention allows defective cells or defective sectors to be remapped as soon as they are detected thereby enabling error correction codes to adequately rectify the relatively few errors that may crop up in the system.

10 According to yet another aspect of the present invention, a write cache is used to minimize the number of writes to the Flash EEPROM memory. In this way the Flash EEPROM memory will be subject to fewer stress inducing write/erase cycles, thereby retarding its aging. The most active data files are written to the
15 cache memory instead of the Flash EEPROM memory. Only when the activity levels have reduced to a predetermined level are the data files written from the cache memory to the Flash EEPROM memory. Another advantage of the invention is the increase in write throughput by virtue
20 of the faster cache memory.

According to yet another aspect of the present invention, one or more printed circuit cards are provided which contain controller and EEPROM circuit chips for use in a computer system memory for long term,
25 non-volatile storage, in place of a hard disk system, and which incorporate various of the other aspects of this invention alone and in combination.

Additional objects, features, and advantages of the present invention will be understood from the
30 following description of its preferred embodiments, which description should be taken in conjunction with the accompanying drawings.

466227-8646680

5

Brief Description of the Drawings

Fig. 1A is a general microprocessor system including the Flash EEprom memory system of the present invention;

5 Fig. 1B is schematic block diagram illustrating a system including a number of Flash EEprom memory chips and a controller chip;

Fig. 2 is a schematic illustration of a system of Flash EEprom chips, among which memory sectors are selected to be erased; on a Flash EEprom chip

10 Fig. 3A is a block circuit diagram (in the controller) for implementing selective multiple sector erase according to the preferred embodiment;

Fig. 3B shows details of a typical register used to select a sector for erase as shown in Fig. 2A;

Fig. 4 is a flow diagram illustrating the erase sequence of selective multiple sector erase;

Fig. 5 is a schematic illustration showing the partitioning of a Flash EEprom sector into a data area and a spare redundant area;

20 Fig. 6 is a circuit block diagram illustrating the data path control during read operation using the defect mapping scheme of the preferred embodiment;

Fig. 7 is a circuit block diagram illustrating the data path control during the write operation using the defect mapping scheme of the preferred embodiment;

25 Fig. 8 is a block diagram illustrating the write cache circuit inside the controller.

20050499 8646680

2

EEprom System

One aspect of the present invention is the substitution of a specific type of semiconductor memory system for the disk drive but without having to sacrifice non-volatility, ease of erasing and rewriting data into the memory, speed of access, low cost and reliability. This is accomplished by employing an array of electrically erasable programmable read only memories (EEPROM's) integrated circuit chips. This type of memory has additional advantages of requiring less power to operate, and of being lighter in weight than a hard disk drive magnetic media memory, thereby being especially suited for battery operated portable computers.

7

serial data line 35. Similarly, data and status signals are communicated from the EEprom 33 to the controller 31 over serial data lines 37. Other control and status circuits between the controller 31 and the EEprom array 33 are not shown in Figure 1A.

Referring to Figure 1B, the controller 31 is preferably formed primarily on a single integrated circuit chip. It is connected to the system address and data bus 39, part of the system bus 33, as well as being connected to system control lines 41, which include interrupt, read, write and other usual computer system control lines.

The EEprom array 33 includes a number of EEprom integrated circuit chips 43, 45, 47, etc. Each includes a respective chip select and enable line 49, 51 and 53 from interface circuits 40. The interface circuits 40 also act to interface between the serial data lines 35, 37 and a circuit 55. Memory location addresses and data being written into or read from the EEprom chips 43, 45, 47, etc. are communicated from a bus 55, through logic and register circuits 57 and thence by another bus 59 to each of the memory chips 43, 45, 47 etc.

The bulk storage memory 29 of Figures 1A and 1B can be implemented on a single printed circuit card for moderate memory sizes. The various lines of the system buses 39 and 41 of Figure 1B are terminated in connecting pins of such a card for connection with the rest of the computer system through a connector. Also connected to the card and its components are various standard power supply voltages (not shown).

For large amounts of memory, that which is conveniently provided by a single array 33 may not be enough. In such a case, additional EEprom arrays can be connected to the serial data lines 35 and 37 of the controller chip 31, as indicated in Figure 1B. This is

06622T" 86466680

5

Erase of Memory Structures

10

20

30

approach.

In the present invention, the Flash EEPROM memory is divided into sectors where all cells within each sector are erasable together. Each sector can be addressed separately and selected for erase. One important feature is the ability to select any combination of sectors for erase together. This will allow for a much faster system erase than by doing each one independently as in prior art.

Figure 2 illustrates schematically selected multiple sectors for erase. A Flash EEPROM system includes one or more Flash EEPROM chips such as 201, 203, 205. They are in communication with a controller 31 through lines 209. Typically, the controller 31 is itself in communication with a microprocessor system (not shown). The memory in each Flash EEPROM chip is partitioned into sectors where all memory cells within a sector are erasable together. For example, each sector may have 512 byte (i.e. 512x8 cells) available to the user, and a chip may have 1024 sectors. Each sector is individually addressable, and may be selected, such as sectors 211, 213, 215, 217 in a multiple sector erase. As illustrated in figure 2, the selected sectors may be confined to one EEPROM chip or be distributed among several chips in a system. The sectors that were selected will all be erased together. This capability will allow the memory and system of the present invention to operate much faster than the prior art architectures.

Figure 3A illustrates a block diagram circuit 220 on a Flash EEPROM chip (such as the chip 201 of figure 2) with which one or more sectors such as 211, 213 are selected (or deselected) for erase. Essentially, each sector such as 211, 213 is selected or tagged by setting the state of an erase enable register

00999499 12999

10

20

30

35

erasing in line 209. The device will then erase all the sectors that have been selected (i.e. the sectors 211 and 213) at one time. In addition to erasing the desired sectors within a chip, the architecture of the present system permits selection of sectors across various chips for simultaneous erase.

Figures 4(1)-4(11) illustrate the algorithm used in conjunction with the circuit 220 of figure 3A. In figure 4(1), the controller will shift the address into the circuit 220 which is decoded in the line to the erase enable register associated with the sector that is to be erased. In figure 4(2), the controller shifts in a command that is decoded to a set erase enable command which is used to latch the address decode signal onto the erase enable register for the addressed sector. This tags the sector for subsequent erase. In figure 4(3), if more sectors are to be tagged, the operations described relative to figures 4(1)-4(2) are repeated until all sectors intended for erase have been tagged. After all sectors intended for erase have been tagged, the controller initiates an erase cycle as illustrated in figure 4(4).

Optimized erase implementations have been disclosed in two copending U.S. patent applications. They are copending U.S. patent applications, Serial No. 204,175, filed June 8, 1988, by Dr. Eliyahou Harari, and one entitled "Multi-State EEprom Read and Write Circuits and Techniques," ^{new patent no. 5,095,344} filed on the same day as the present application, by Sanjay Mehrotra and Dr. Eliyahou Harari. The disclosures of the two applications are hereby incorporate by reference. The Flash EEprom cells are erased by applying a pulse of erasing voltage followed by a read to verify if the cells are erased to the "erased" state. If not, further pulsing and verifying

165227-86466680

a

a
a A₂

are repeated until the cells are verified to be erased. By erasing in this controlled manner, the cells are not subject to over-erasure which tends to age the EEprom device prematurely as well as make the cells harder to program.

As the group of selected sectors is going through the erase cycle, some sectors will reach the "erase" state earlier than others. Another important feature of the present invention is the ability to remove those sectors that have been verified to be erased from the group of selected sectors, thereby preventing them from over-erasing.

Returning to figure 4(4), after all sectors intended for erase have been tagged, the controller initiates an erase cycle to erase the group of tagged sectors. In figure 4(5), the controller shifts in a global command called Enable Erase into each Flash EEprom chip that is to perform an erase. This is followed in figure 4(5) by the controller raising of the erase voltage line (Ve) to a specified value for a specified duration. The controller will lower this voltage at the end of the erase duration time. In figure 4(6), the controller will then do a read verify sequence on the sectors selected for erase. In figure 4(7), if none of the sectors are verified, the sequences illustrated in figures 4(5)-4(7) are repeated. In figures 4(8) and 3(9), if one or more sectors are verified to be erased, they are taken out of the sequence. Referring also to figure 3A, this is achieved by having the controller address each of the verified sectors and clear the associated erase enable registers back to a LOW with a clear enable command in bus 237. The sequences illustrated in figures 4(5)-4(10) are repeated until all the sectors in the group are verified to be erased in figure 4(11). At the completion of the

0899498-122997

20

30

By doing this in a global way to all the chips, less time will be taken to reset all the erase enable registers.

5 An additional performance capability is to have the ability to do erase operations without regard to chip select. Once an erase is started in some of the memory chips, the controller in the system can access other memory chips and do read and write operations on them. In addition, the device(s) doing the erase can be
10 selected and have an address loaded for the next command following the erase.

Defect Mapping

Physical defects in memory devices give rise to hard errors. Data becomes corrupted whenever it is
15 stored in the defective cells. In conventional memory devices such as RAM's and Disks, any physical defects arising from the manufacturing process are corrected at the factory. In RAM's, spare redundant memory cells on chip may be patched on, in place of the defective cells.
20 In the traditional disk drive, the medium is imperfect and susceptible to defects. To overcome this problem manufacturers have devised various methods of operating with these defects present, the most usual being defect mapping of sectors. In a normal disk system the media
25 is divided into cylinders and sectors. The sector being the basic unit in which data is stored. When a system is partitioned into the various sectors the sectors containing the defects are identified and are marked as bad and not to be used by the system. This is done in
30 several ways. A defect map table is stored on a particular portion of the disk to be used by the interfacing controller. In addition, the bad sectors are marked as bad by special ID and flag markers. When the defect is addressed, the data that would normally be

0899498-12997
26227-8646880

stored there is placed in an alternative location. The requirement for alternative sectors makes the system assign spare sectors at some specific interval or location. This reduces the amount of memory capacity and is a performance issue in how the alternative sectors are located.

One important application of the present invention is to replace a conventional disk storage device with a system incorporating an array of Flash EEprom memory chips. The EEprom system is preferably set up to emulate a conventional disk, and may be regarded as a "solid-state disk".

In a "disk" system made from such solid-state memory devices, low cost considerations necessitate efficient handling of defects. Another important feature of the invention enables the error correction scheme to conserve as much memory as possible. Essentially, it calls for the defective cells to be remapped cell by cell rather than by throwing away the whole sector (512 bytes typically) whenever a defect occurs in it. This scheme is especially suited to the Flash EEprom medium since the majority of errors will be bit errors rather than a long stream of adjacent defects as is typical in traditional disk medium.

In both cases of the prior art RAM and magnetic disk, once the device is shipped from the factory, there is little or no provision for replacing hard errors resulting from physical defects that appear later during normal operation. Error corrections then mainly rely on schemes using error correction codes (ECC).

The nature of the Flash EEprom device predicates a higher rate of cell failure especially with increasing program/erase cycling. The hard errors that accumulate with use would eventually overwhelm the ECC and render the device unusable. One important feature

00559498-125997

16

of the present invention is the ability for the system to correct for hard errors whenever they occur. Defective cells are detected by their failure to program or erase correctly. Also during read operation, defective cells are detected and located by the ECC. As soon as a defective cell is identified, the controller will apply defect mapping to replace the defective cell with a space cell located usually within the same sector. This dynamic correction of hard errors, in addition to conventional error correction schemes, significantly prolongs the life of the device.

Another feature of the present invention is an adaptive approach to error correction. Error correction code (ECC) is employed at all times to correct for soft errors as well as any hard errors that may arise. As soon as a hard error is detected, defect mapping is used to replace the defective cell with a spare cell in the same sector block. Only when the number of defective cells in a sector exceeds the defect mapping's capacity for that specific sector will the whole sector be replaced as in a conventional disk system. This scheme minimized wastage without compromising reliability.

Figure 5 illustrates the memory architecture for the cell remapping scheme. As described before, the Flash EEPROM memory is organized into sectors where the cells in each sector are erasable together. The memory architecture has a typical sector 401 organized into a data portion 403 and a spare (or shadow) portion 405. The data portion 403 is memory space available to the user. The spare portion 405 is further organized into an alternative defects data area 407, a defect map area 409, a header area 411 and an ECC and others area 413. These areas contain information that could be used by the controller to handle the defects and other overhead information such as headers and ECC.

0899498-122997

5

10

20

30

port 505. Then the microprocessor loads a DMA controller 507 with the starting location in buffer memory or bus address that the data read should be sent. Then the microprocessor loads the header information (Head, Cylinder and sector) into a holding register file 509. Finally, the microprocessor loads a command sequencer 511 with a read command before passing control to the controller 31.

After assuming control, the controller 31 first addresses the header of the sector and verifies that the memory is accessed at the address that the user had specified. This is achieved by the following sequence. The controller selects a memory chip (chip select) among the memory device 33 and shifts the address for the header area from the address generator 503 out to the selected memory chip in the memory device 33. The controller then switches the multiplexer 513 and shifts also the read command out to the memory device 33. Then the memory device reads the address sent it and begins sending serial data from the addressed sector back to the controller. A receiver 515 in the controller receives this data and puts it in parallel format. In one embodiment, once a byte (8 bits) is compiled, the controller compares the received data against the header data previously stored by the microprocessor in the holding register file 509. If the compare is correct, the proper location is verified and the sequence continues.

Next the controller 31 reads the defect pointers and loads these bad address locations into the holding register file 509. This is followed by the controller reading the alternative defects data that were written to replace the bad bits as they were written. The alternative bits are stored in an alternative defects data file 517 that will be accessed

08090498-122997

Once the Header has been determined to be a match and the defect pointers and alternative bits have been loaded, the controller begins to shift out the address of the lowest address of the desired sector to be read. The data from the sector in the memory device 33 is then shifted into the controller chip 31. The receiver 515 converts the data to a parallel format and transfers each byte into a temporary holding FIFO 519 to be shipped out of the controller.

A pipeline architecture is employed to provide efficient throughput as the data is gated through the controller from the receiver 515 to the FIFO 519. As each data bit is received from memory the controller is comparing the address of the data being sent (stored in the address generator 507) against the defect pointer map (stored in the register file 509). If the address is determined to be a bad location, by a match at the output of the comparator 521, the bad bit from the memory received by the receiver 515 is replaced by the good bit for that location. The good bit is obtained from the alternative defects data file 517. This is done by switching the multiplexer 523 to receive the good bit from the alternative defects data file instead of the bad bit from the receiver 515, as the data is sent to the FIFO 519. Once the corrected data is in the FIFO it is ready to be sent to buffer memory or system memory (not shown). The data is sent from the controller's FIFO 519 to the system memory by the controller's DMA controller 507. This controller 507 then requests and gets access to the system bus and puts out an address and gates the data via an output interface 525 out to the system bus. This is done as each byte gets loaded into the FIFO 519. As the corrected data is loaded into the FIFO it will also be

gated into an ECC hardware 527 where the data file will be acted on by the ECC.

Thus in the manner described, the data read from the memory device 33 is gated through the controller 31 to be sent to the system. This process continues until the last bit of addressed data has been transferred.

In spite of defect mapping of previously detected defective cells, new hard errors might occur since the last mapping. As the dynamic defect mapping constantly "puts away" new defective cells, the latest hard error that may arise between defect mapping would be adequately handled by the ECC. As the data is gated through the controller 31, the controller is gating the ECC bits into the ECC hardware 527 to determine if the stored value matched the just calculated remainder value. If it matches then the data transferred out to the system memory was good and the read operation was completed. However, if the ECC registers an error then a correction calculation on the data sent to system memory is performed and the corrected data retransmitted. The method for calculating the error can be done in hardware or software by conventional methods. The ECC is also able to calculate and locate the defective cell causing the error. This may be used by the controller 31 to update the defect map associated with the sector in which the defective cell is detected. In this manner, hard errors are constantly removed from the Flash EEprom system.

Figure 7 illustrates the write data path control in the preferred embodiment. The first portion of a write sequence is similar to a read sequence described previously. The microprocessor first loads the Address pointers for the memory device 33 and the DMA as in the read sequence. It also loads the header

00000000-122997

desired into the address generator 503 and the command queue into the command sequencer 511. The command queue is loaded with a read header command first. Thereafter, control is passed over to the controller 31. The controller then gates the address and command to the memory device 33, as in the read sequence. The memory device returns header data through controller's receiver 515. The controller compares the received header data to the expected value (stored in the holding register file 509). If the compare is correct, the proper location is verified and the sequence continues. Then the controller loads the defective address pointers from the memory device 33 into the holding register file 509 and the alternative data into the alternative defects data file 517.

Next, the controller begins to fetch the write data from system memory (not shown). It does this by getting access to the system bus, outputs the memory or bus address and does the read cycle. It pulls the data into a FIFO 601 through an input interface 603. The controller then shifts the starting sector address (lowest byte address) from the address generator 503 to the selected memory device 33. This is followed by data from the FIFO 601. These data are routed through multiplexers 605 and 513 and converted to serial format before being sent to the memory device 33. This sequence continues until all bytes for a write cycle have been loaded into the selected memory.

A pipeline architecture is employed to provide efficient throughput as the data is gated from the FIFO 601 to the selected memory 33. The data gated out of the FIFO 601 is sent to the ECC hardware 527 where a remainder value will be calculated within the ECC. In the next stage, as the data is being sent to the memory device through multiplexers 605 and 513, the comparator

0899498-12997

521 is comparing its address from the address generator 503 to the defect pointer address values in the holding register file 509. When a match occurs, indicating that a defective location is about to be written, the controller saves this bit into the alternative defect data file 517. At the same time, all bad bits sent to memory will be sent as zeroes.

After the bytes for a write cycle have been loaded into the selected memory device, the controller issues a program command to the memory device and initiate a write cycle. Optimized implementations of write operation for Flash EEPROM device have been disclosed in two previously cited co-pending U.S. patent applications, Serial No. ^{now patent no. 5,095,344} 204,175, and one entitled "Multi-State EEPROM Read and Write Circuits and Techniques." Relevant portions of the disclosures are hereby incorporated by reference. Briefly, during the write cycle, the controller applies a pulse of programming (or writing) voltages. This is followed by a verify read to determine if all the bits have been programmed properly. If the bits did not verify, the controller repeats the program/verify cycle until all bits are correctly programmed.

If a bit fails to verify after prolonged program/verify cycling, the controller will designate that bit as defective and update the defect map accordingly. The updating is done dynamically, as soon as the defective cell is detected. Similar actions are taken in the case of failure in erase verify.

After all the bits have been programmed and verified, the controller loads the next data bits from the FIFO 601 and addresses the next location in the addressed sector. It then performs another program/verify sequence on the next set of bytes. The sequence continues until the end of the data for that

0399498-12299
265227-8646680

sector. Once this has occurred, the controller addresses the shadow memory (header area) associated with the sector (see figure 5) and writes the contents of the ECC registers into this area.

5 In addition, the collection of bits that was flagged as defective and were saved in the alternative defects data file 516 is then written in memory at the alternative defects data locations (see figure 5), thereby saving the good bit values to be used on a
10 subsequent read. Once these data groups are written and verified, the sector write is considered completed.

The present invention also has provision for defect mapping of the whole sector, but only after the number of defective cells in the sector has exceeded the
15 cell defect mapping's capacity for that specific sector. A count is kept of the number of defective cells in each sector. When the number in a sector exceeds a predetermined value, the controller marks that sector as defective and maps it to another sector. The defect
20 pointer for the linked sectors may be stored in a sector defect map. The sector defect map may be located in the original defective sector if its spare area is sufficiently defect-free. However, when the data area of the sector has accumulated a large number of defects,
25 it is quite likely that the spare area will also be full of defects.

Thus, it is preferable in another embodiment to locate the sector map in another memory maintained by the controller. The memory may be located in the
30 controller hardware or be part of the Flash EEPROM memory. When the controller is given an address to access data, the controller compares this address against the sector defect map. If a match occurs then access to the defective sector is denied and the
35 substitute address present in the defect map is entered,

0899498-125997
156227-8646680

and the corresponding substitute sector is accessed instead.

In yet another embodiment, the sector remapping is performed by the microprocessor. The microprocessor
 5 looks at the incoming address and compares it against the sector defect map. If a match occurs, it does not issue the command to the controller but instead substitute the alternative location as the new command.

Apart from the much higher speed of the solid-
 10 state disk, another advantage is the lack of mechanical parts. The long seek times, rotational latency inherent in disk drives are not present. In addition, the long synchronization times, sync mark detects and write gaps are not required. Thus the overhead needed for
 15 accessing the location where data is to be read or written is much less. All of these simplifications and lack of constraints result in a much faster system with much reduced overheads. In addition, the files can be arranged in memory in any address order desired, only
 20 requiring the controller to know how to get at the data as needed.

Another feature of the invention is that defect mapping is implemented without the need to interrupt the data stream transferred to or from the sector. The data
 25 in a block which may contain errors are transferred regardless, and is corrected afterwards. Preserving the sequential addressing will result in higher speed by itself. Further, it allows the implementation of an efficient pipeline architecture in the read and write
 30 data paths.

Write Cache System

Cache memory is generally used to speed up the performance of systems having slower access devices. For example in a computer system, access of data from

0899498-12597

25

5

15

35

a
a

by excessive program/erase cycles. Even with the improved Flash EEPROM memory device as disclosed in co-pending U.S. patent applications, ^{now patent no. 5,095,344,} Serial No. 204,175, and one entitled "Multi-state EEPROM Read and Write Circuits and Techniques," by Sanjay Mehrotra and Dr. Eliyahou Harari filed on the same day as the present application, the endurance limit is approximately 10^6 program/erase cycles. In a ten-year projected life time of the device, this translates to a limit of one program/erase cycle per 5 minutes. This may be marginal in normal computer usage.

456227-86465530

To overcome this problem, a cache memory is used in a novel way to insulate the Flash EEPROM memory device from enduring too many program/erase cycles. The primary function of the cache is to act on writes to the Flash EEPROM memory and not on reads of the Flash EEPROM memory, unlike the case with traditional caches. Instead of writing to the Flash EEPROM memory every time the data is updated, the data may be operated on several times in the cache before being committed to the Flash EEPROM memory. This reduces the number of writes to the Flash EEPROM memory. Also, by writing mostly into the faster cache memory and reducing the number of writes to the slower Flash EEPROM, an additional benefit is the increase in system write throughput.

A relatively small size cache memory is quite effective to implement the present invention. This helps to overcome the problem of data loss in the volatile cache memory during a power loss. In that event, it is relatively easy to have sufficient power reserve to maintain the cache memory long enough and have the data dumped into a non-volatile memory such as a specially reserved space in the Flash EEPROM memory. In the event of a power down or and power loss to the system, the write cache system may be isolated from the

27

system and a dedicated rechargeable power supply may be switch in only to power the cache system and the reserved space in the Flash EEprom memory.

Figure 8 illustrates schematically a cache system 701 as part of the controller, according to the present invention. On one hand the cache system 701 is connected to the Flash EEprom memory array 33. On the other hand it is connected to the microprocessor system (not shown) through a host interface 703. The cache system 701 has two memories. One is a cache memory 705 for temporarily holding write data files. The other is a tag memory 709 for storing relevant information about the data files held in the cache memory 705. A memory timing/control circuit 713 controls the writing of data files from the cache memory 705 to the Flash EEprom memory 33. The memory control circuit 713 is responsive to the information stored in the tag memory as well as a power sensing input 715 which is connected through the host interface 703 via a line 717 to the power supply of the microprocessor system. A power loss in the microprocessor system will be sensed by the memory control circuit 713 which will then download all the data files in the volatile cache memory 705 to the non-volatile Flash EEprom memory 33.

In the present invention, the Flash EEprom memory array 33 is organized into sectors (typically 512 byte size) such that all memory cells within each sector are erasable together. Thus each sector may be considered to store a data file and a write operation on the memory array acts on one or more such files.

During read of a new sector in the Flash EEprom memory 33, the data file is read out and sent directly to the host through the controller. This file is not used to fill the cache memory 705 as is done in the traditional cache systems.

0399498-122997

After the host system has processed the data within a file and wishes to write it back to the Flash EEprom memory 33, it accesses the cache system 701 with a write cycle request. The controller then intercepts this request and acts on the cycle.

In one embodiment of the invention, the data file is written to the cache memory 705. At the same time, two other pieces of information about the data file are written to a tag memory 709. The first is a file pointer which identifies the file present in the cache memory 705. The second is a time stamp that tells what time the file was last written into the cache memory. In this way, each time the host wishes to write to the Flash EEprom memory 33, the data file is actually first stored in the cache memory 705 along with pointers and time stamps in the tag memory 709.

In another embodiment of the invention, when a write from the host occurs, the controller first checks to see if that file already existed in the cache memory 705 or has been tagged in the tag memory 709. If it has not been tagged, the file is written to the Flash memory 33, while its identifier and time stamp are written to the tag memory 709. If the file already is present in the cache memory or has been tagged, it is updated in the cache memory and not written to the Flash memory. In this way only infrequently used data files are written into the Flash memory while frequently used data files are trapped in the cache memory.

In yet another embodiment of the invention, when a write from the host occurs, the controller first checks to see if that data file has been last written anywhere within a predetermined period of time (for example, 5 minutes). If it has not, the data file is written to the Flash memory 33, while its identifier and time stamp are written to the tag memory 709. If

46522T 8646680

5

10

20

25

30

35

Two things can happen to a file's time stamp, depending on the activity of the file. One possibility is for the time stamp to be reset in the event of a new activity occurring. The other possibility is that no new activity occurs for the file and the time stamp continues to increment until the file is removed from the cache. In practice a maximum limit may be reached if the time stamp is allowed to increase indefinitely. For example, the system may allow the time stamp to increment to a maximum period of inactivity of 5 minutes. Thus, when a data file is written in the cache memory, the time stamp for the file is set at its initial value. Then the time stamp will start to age, incrementing at every time step unless reset to its initial value again by another write update. After say, 5 minutes of inactivity, the time stamp has incremented to a maximum terminal count.

In one embodiment of keeping count, a bit can be shifted one place in a shift register each time a count increment for a file occurs. If the file is updated (a new activity has occurred) the bit's location will be reset to the initial location of the shift register. On the other hand, if the file remains inactive the bit will eventually be shifted to the terminal shift position. In another embodiment, a count value for each file is stored and incremented at each time step. After each increment, the count value is compared to a master counter, the difference being the time delay in question.

Thus, if a file is active its incremented time stamp is reset back to the initial value each time the data file is rewritten. In this manner, files that are constantly updated will have low time stamp identifiers and will be kept in cache until their activity decreases. After a period of inactivity has expired,

0399498-122997

they acquire the maximum time stamp identifiers. The inactive files are eventually archived to the Flash memory freeing space in the cache memory for new, more active files. Space is also freed up in the tag memory
5 when these inactive files are moved to the Flash memory.

At any time when room must be made available for new data files coming into the cache memory, the controller removes some of the older files and archives them to the Flash memory 33. Scheduling is done by a
10 memory timing/control circuit 713 in the controller. The decision to archive the file is based on several criteria. The controller looks at the frequency of writes occurring in the system and looks at how full the cache is. If there is still room in the cache, no
15 archiving need to be done. If more room is needed, the files with the earliest time stamps are first removed and archived to the Flash memory.

Although the invention has been described with implementation in hardware in the controller, it is to
20 be understood that other implementations are possible. For example, the cache system may be located elsewhere in the system, or be implemented by software using the existing microprocessor system. Such variations are within the scope of protection for the present
25 invention.

The Profile of how often data is written back to the Flash memory is determined by several factors. It depends on the size of the cache memory and the frequency of writes occurring in the system. With a
30 small cache memory system, only the highest frequency files will be cached. Less frequently accessed files will also be cached with increasing cache memory size. In the present invention, a relatively cheap and small amount of cache memory, preferably about 1 Mbyte, may be
35 used to good advantage. By not constantly writing the

009949 12297
46522T 846680

most active files (the top 5%), the write frequency of the Flash EEPROM may be reduced from the usual one every millisecond to one every 5 minutes. In this way the wear-out time for the memory can be extended almost indefinitely. This improvement is also accompanied by increased system performance during write.

Incorporating time tag into the write cache concept has the advantage that the size of the write cache buffer memory can be relatively small, since it is used only to store frequently written data files, with all other files written directly into the Flash EEPROM memory. A second advantage is that the management of moving data files in and out of the write cache buffer can be automated since it does not require advanced knowledge of which data files are to be called next.

The various aspects of the present invention that have been described co-operate in a system of Flash EEPROM memory array to make the Flash EEPROM memory a viable alternative to conventional non-volatile mass storage devices.

While the embodiments of the various aspects of the present invention that have been described are the preferred implementation, those skilled in the art will understand that variations thereof may also be possible. Therefore, the invention is entitled to protection within the full scope of the appended claims.

46522T-8546680